CLAIM SET AS AMENDED

1. (Currently Amended) A method for processing data expressed as a class in ~~JAVA~~ object-oriented programming language to produce data expressed as an extensible markup language (XML) document, comprising:

loading a named ~~JAVA~~ object-oriented programming language class;

determining if the loaded ~~JAVA~~ object-oriented programming language class implements a predefined interface, said predefined interface comprising annotations including a first parameter for associating a field of said ~~JAVA~~ object-oriented programming language class with a corresponding XML element tag; a second parameter for specifying ~~a JAVA~~ an object-oriented programming language class to be instantiated when constructing said ~~JAVA~~ object-oriented programming language class field from said XML file; a third parameter; for identifying ~~a JAVA~~ an object-oriented programming language method to invoke for retrieving said ~~JAVA~~ object-oriented programming language class field, and a fourth parameter for identifying ~~a JAVA~~ an object-oriented programming language method to invoke for retrieving this method; and

in the case of said loaded ~~JAVA~~ object-oriented programming language class, implementing said predefined interface iteratively processing each field descriptor within the loaded ~~JAVA~~ object-oriented programming language class to retrieve a corresponding XML tag; and

transferring field values to new elements created using said corresponding XML tags.

2. (Currently Amended) A method for processing data expressed as an extensible markup language (XML) document to produce a data expresses as a class in ~~JAVA~~ object-oriented programming language, comprising:

instantiating an object of a desired ~~JAVA~~ object-oriented programming language class;

in the case of said instantiated object, implementing a predefined interface, iteratively processing each object included within said instantiated object according to the steps of:

retrieving field descriptors associated with said object being processed;

creating an object of a specified ~~JAVA~~ object-oriented programming language type for each XML element corresponding to a field descriptor; and

storing the created object in the currently processed object.

3. (Currently Amended) A method for adapting an object in ~~JAVA~~ object-oriented programming language to an application programming interface (API) for converting said ~~JAVA~~ object-oriented programming language object to data expressed in eXtensible Markup Language (XML), comprising:

annotating said ~~JAVA~~ object-oriented programming language object to be converted to said XML to ~~including~~ include the steps of:

identifying a respective XML tag ;

identifying ~~a JAVA~~ an object-oriented programming language class to be instantiated when constructing said ~~JAVA~~ object-oriented programming language object field from an XML file ;

identifying ~~a JAVA~~an object-oriented programming language method to invoke for retrieving said ~~JAVA~~ object-oriented programming language object; and

identifying ~~a JAVA~~ an object-oriented programming language method to invoke for retrieving said retrieval method.

4. (Currently Amended) A method for converting at least ~~date~~ data from ~~JAVA~~ object-oriented programming language to data in extensible mark-up language (XML) using an application programming interface (API), the method comprising the steps of:

retrieving a field description;

determining ~~JAVA~~ object-oriented programming language conversion parameters by examining an annotation associated with each ~~JAVA~~ object-oriented programming language

element to be converted to XML, said annotation defining for each ~~JAVA~~ object-oriented programming language element at least a corresponding XML tag, a corresponding object class, a corresponding field retrieval method, and a corresponding method retrieval method.

5. (Currently Amended) The method of claim 4, further comprising the steps of:

loading a named ~~JAVA~~ object-oriented programming language class;

determining if a loaded ~~JAVA~~ object-oriented programming language class implements a predefined interface, said predefined interface comprising annotations including:

a first parameter for associating a field of said ~~JAVA~~ object-oriented programming language class with a corresponding XML element tag;

a second parameter for specifying ~~a JAVA~~ an object-oriented programming language class to be instantiated when constructing said ~~JAVA~~ object-oriented programming language class field from said XML file;

a third parameter for identifying ~~a JAVA~~ an object-oriented programming language method to invoke for retrieving said ~~JAVA~~ object-oriented programming language class field; and

a fourth parameter, for identifying ~~a JAVA~~ an object-oriented programming language method to invoke for retrieving this method; and

in the case of said loaded ~~JAVA~~ object-oriented programming language class, implementing said predefined interface iteratively processing each field descriptor within the loaded ~~JAVA~~ object-oriented programming language class to retrieve the corresponding XML tag; and

transferring field values to new elements created using said corresponding XML tags.

6. (Currently Amended) The method of claim 4, further comprising:

instantiating an object of a desired ~~JAVA~~ object-oriented programming language class;

in the case of said instantiated object implementing a predefined interface, iteratively processing each object included within said instantiated object according to the steps of:

retrieving field descriptors associated with an object being processed;

creating an object of a specified ~~JAVA~~ object-oriented programming language type for each XML element corresponding to a field descriptor; and

storing the created object in the currently processed object.

7. (Currently Amended) A method for describing a class field in ~~JAVA~~ object-oriented programming language in a manner facilitating conversion to data expressed in extensible markup language (XML), comprising the steps of:

associating said ~~JAVA~~ object-oriented programming language class field with a corresponding XML element tag with a first parameter;

specifying ~~a JAVA~~ an object-oriented programming language class to be instantiated when constructing said ~~JAVA~~ object-oriented programming language class field from said XML file with a second parameter;

identifying ~~a JAVA~~ an object-oriented programming language method to invoke for retrieving said ~~JAVA~~ object-oriented programming language class field with a third parameter; and

retrieving the ~~JAVA~~ object-oriented programming language class field with a fourth parameter.

8. (Currently Amended) The method of claim 7, further comprising:

specifying a type of ~~JAVA~~ object-oriented programming language object to instantiate for an XML element representing a collection with a fifth parameter.

9. (Previously Presented) The method of claim 8, further comprising:

specifying a tag name to use for each element representing a collection with a sixth parameter.

10. (Previously Presented) The method of claim 8, wherein said collection comprises a HashTable.

11. (Currently Amended) A computer system for executing an application programming interface (API) function for converting data expressed as a class in ~~JAVA~~ object-oriented programming language to data expressed in extensible mark-up language (XML), comprising:

processing means for executing the API function and memory means for storing the data to be converted,

wherein the API function includes a field description retrieval method for determining ~~JAVA~~ object-oriented programming language conversion parameters by examining an annotation associated with each ~~JAVA~~ object-oriented programming language element to be converted to XML, said annotation defining for each ~~JAVA~~ object-oriented programming language element at least a corresponding XML tag, a corresponding object class, a corresponding field retrieval method, and a corresponding method retrieval method.

12. (Currently Amended) The computer system for executing an application programming interface (API) function of claim 11, further comprising to the steps of:

loading a named ~~JAVA~~ object-oriented programming language class into the processor;

determining if the loaded ~~JAVA~~ object-oriented programming language class implements a predefined interface, said predefined interface comprising annotations including a first parameter for associating ~~a JAVA~~ an object-oriented programming language class field with a corresponding XML element tag, a second parameter for specifying ~~a JAVA~~ an object-oriented programming language class to be instantiated when constructing said ~~JAVA~~ object-oriented

programming language class field from said XML file, a third parameter for identifying ~~a JAVA~~ an object-oriented programming language method to invoke for retrieving said ~~JAVA~~ object-oriented programming language class field, and a fourth parameter for identifying ~~a JAVA~~ an object-oriented programming language method to invoke for retrieving this method; and

in the case of said loaded ~~JAVA~~ object-oriented programming language class implementing said predefined interface, iteratively processing each field descriptor within the loaded ~~JAVA~~ object-oriented programming language class to retrieve corresponding XML tag; and

transferring field values to new elements created using said corresponding XML tags to the memory means of the computer system.

13. (Currently Amended) The computer system for executing an application programming interface (API) function of claim 11, further comprising the steps of:

instantiating an object of the desired ~~JAVA~~ object-oriented programming language class;

in the case of said instantiated object implementing a predefined interface, iteratively processing each object included within said instantiated object according to the steps of:

retrieving field descriptors associated with ~~an~~ the object being processed;

creating an object of a specified ~~JAVA~~ object-oriented programming language type for each XML element corresponding to a field descriptor; and

storing the created object in the currently processed object.